Sign in

Google Web Images Video[New!] News Maps more »

filter each row predicate iterator Search | Advanced Search Preferences

Web Results **1** - **10** of about **38,100** for **filter each row predicate iterator**. (**0.54** seconds)

Did you mean: filter each row predicate ***operator***

RDQL (class_rdql.php)
Each member of the array is a **row** of the RDQL result. **Each row** is an asociative array ...
Description: The RDQL engine receives an RDF **iterator** object when ...
cvs.sourceforge.net/viewcvs.py/phpxmlclasses/rdql/class_rdql.html?rev=1.2 - 25k -
Cached - Similar pages

Craig Freedman's WebLog : Scans vs. Seeks
... we read **each row** in the orders table, evaluate the **predicate** "where ... we may evaluate it in a separate **filter iterator**. The residual **predicate** appears ...
blogs.msdn.com/craigfr/archive/2006/06/26/647852.aspx - 29k - Cached - Similar pages

XulPlanet.com - Neil's Place
You cannot make **each row** a different height, but you can change the height ... However, simple rules may also **iterate** over a single **predicate** pointing out ...
www.xulplanet.com/ndeakin/archive/2005/7/ - 77k - Cached - Similar pages

> XulPlanet.com - Neil's Place
> We need to **iterate** over the 'type' **predicate** to find the individual countries. ... will be that the first buttons in **each row** will all have the same label. ...
> www.xulplanet.com/ndeakin/archive/2005/6/ - 80k - Cached - Similar pages
> [More results from www.xulplanet.com]

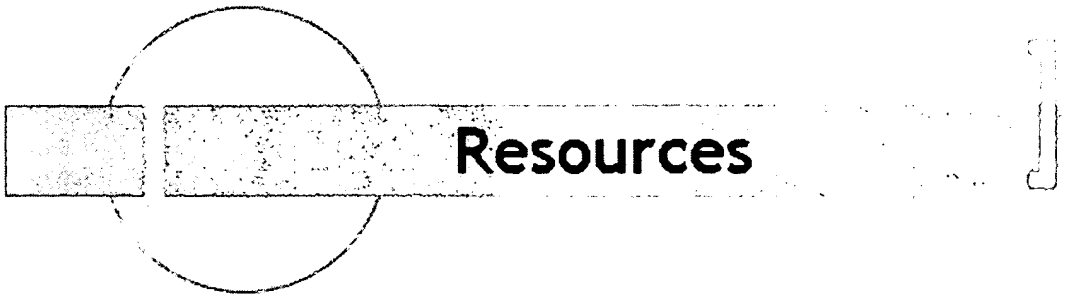
19 Using EXPLAIN PLAN
Oracle compares **each row** of the outer set with **each row** of the inner set, returning rows that ... Similar to **iterator**, but based on an IN -list **predicate**. ...
www.stanford.edu/dept/itss/docs/oracle/10g/server.101/b10752/ex_plan.htm - 132k -
Cached - Similar pages

6.830 Problem Set 2: SimpleDB
In SimpleDB, operators are **iterator** based; **each** operator implements the ... **Filter**: This operator only passes on tuples that pass a **Predicate** that is ...
db.csail.mit.edu/6.830/assignments/ps2.html - 30k - Cached - Similar pages

Using EXPLAIN PLAN
Takes **each row** from a table **row** source and finds the corresponding bitmap from a bitmap index. ... Similar to **iterator**, but based on an IN -list **predicate**. ...
www.lc.leidenuniv.nl/awcourse/oracle/server.920/a96533/ex_plan.htm - 123k -
Cached - Similar pages

Algs (Suvi)
Iterator filter(java.util.**Iterator** from, Function pred) ... The last element of **each row** of the association table is the value to which all the other ...
www.cs.helsinki.fi/group/suvi/javadoc/suvi/util/collections/Algs.html - 100k -
Cached - Similar pages

Genezzo::**Row**::RSTab.pm - **Row** Source TABle tied hash class ...
A **filter** is simply a **predicate** which is applied to every **row** -- rows which ... with the notion of start and stop keys -- the **iterator** only returns the rows ...
search.cpan.org/~jcohen/Genezzo-0.62/lib/Genezzo/**Row**/RSTab.pm - 14k -
Cached - Similar pages



Home
Resources
Working Groups
Company

Genezzo Feature Summary

- Multi-user, Multi-server distributed database
- Central web-based cluster administration tool
- Row-level locking
- Transactional Integrity with Level 3 Isolation
- Open Index API
- Online hot backup
- Online Replication
- Parallel Query
- SQL Skins (ANSI SQL module only) with optional compatibility alerts
- Sub-queries
- Estimation of query completion time
- Partial query results
- Very Large Database (VLDB) Support
- Large Data Types - typed BLOB columns via extensible block types
- Platform Requirements: POSIX compliance, Perl 5.6

Genezzo FAQ

How is Genezzo better than the relational databases currently in use?

Genezzo aims to offer a unique combination of cost-effectiveness and industrial strength. While there are a number of proven relational databases in widespread commercial use, the cost of ownership of such products (licensing, installing, administering, and provisioning hardware) has not scaled down at the rate that many other technologies have recently (e.g. web servers, application servers, programming IDE's). Conversely, there are a number of "partial" databases that are available via open source venues that lack the scalability and breadth of application that a commercial database from Oracle or IBM might supply. Moreover, developer and administrator productivity on all existing relational databases is inferior to that of Genezzo users.

Genezzo is designed from the ground up to be a relational database that is distributed across many blade servers, clusters, SMP systems, or servers located in different physical locations. The monolithic and centralized "big iron"

approach to most database installations, even when installed across clusters, does not begin to approach the productivity of the theoretical capacity of a set of servers cooperating to solve a problem in an optimal way. Moreover, replication and distribution of load in Genezzo is nearly turnkey - meaning that actual availability and performance of a Genezzo cluster, as seen by the whole of the Internet, can be tuned by adding machines to "scale out" a service. When a person in California sees acceptable reliability and response time, but a person in Holland does not - is that real uptime? When an administration and operations team has to take the whole business down for a database server upgrade - is that an efficient use of resources?

Is there something architecturally different about this database that allows these problems to be approached?

Absolutely. The architectural goals of Genezzo are openness, extensibility, flexibility, and support for parallel and distributed operation.

Genezzo does more than just user-defined functions or data types. It provides novel access methods and data block definitions as part of a distributed data framework. On-the-fly extension of Genezzo, application upgrades, and Genezzo system upgrades are supported at run-time. Open block formats reduce data integration and upload headaches by allowing many more storage modules to be easily written and plugged into Genezzo's framework.

How is Genezzo so cost-effective and easy to manage?

Genezzo's operational goals are to be scaleable, reliable, manageable. These goals are achieved through the unique architectural investments detailed above.

- scaleable: take advantage of parallel and distributed architecture.
- reliable: provide transaction support for ACID properties on distributed platform.
- manageable: simple, online cluster management.

For example, the ease and deterministic nature of adding a new node into a Genezzo cluster to increase capacity is unparalleled. A Genezzo cluster can be made up of existing hardware investments with differing CPU, memory, and storage allocations. The database auto-tunes itself to distribute the load according to server capabilities as measured by Genezzo doing database work on each server as it is inserted into the cluster. These performance-related settings, although they are visible and overridable, are not driven, as they are with other databases, by a myriad of administrative settings and configuration files. Similarly, removing a node for maintenance can be done with minor impact, affording the opportunity for incremental upgrades of hardware and operating systems with Zero Cluster Down Time.

Genezzo's parallel execution capabilities, used primarily for quorum-based transaction processing, can also be used to maintain hot standby servers and replicant servers that can be swapped in for a primary transactional server.

Transactions are journaled and played back on other Genezzo servers with a configurable delay. Running behind by 5 minutes assured an acceptable level of business continuity in many instances, but when taking a machine out of a cluster, that delay is pushed to zero by Genezzo's automated Replicator process.

Manageability is achieved via a web-based cluster management console that makes administering large, physically dispersed sets of servers approximately the same amount of work as administering a single-machine "starter cluster". Through a good deal of auto-configuration, adding new nodes and performing machine and network maintanance is easy using this scheme. Online upgrades, backups, and schema changes produce the high availability that eliminates much of the issues associated with tasks that would otherwise result in planned downtimes. Rather than managing a single node for optimal performance, Genezzo offers a management model that scales out as needs increase.

How is Genezzo so cost-effective and productive as a development platform?

The notion of a database microkernel that does high-granularity partitioning means a whole system can be compartmentalized into separate subsystems for each development and QA scenario. Through Genezzo's virtual copy semantics, the conceptual and actual costs of creating a new working context for each development, testing, and production scenario are greatly reduced.

Extending Genezzo can be rewarding and, more importantly, does not require deep "database guru" skills. One need not understand the whole system - as each part is simple and operates fairly independently. For example, the data layer's job is to store and retrieve blocks while maintaining block consistency. The SQL layer operates on collections of blocks. A table is a union of a set of blocks (relations). These simple constructs make components within Genezzo easily replacable.

Another unique Genezzoism is deep aggregation, the ability to push processing down to the block layer reduces the volume of data that traverses the operator tree. Aggregative functions such as COUNT, MIN, MAX, AVERAGE, do not require full table scans, as intermediate results are cached and invalidated at the block level (a technique not employed by common RDBMS efforts to date). As an example, each Genezzo block operation can individually filter and count rows, returning a single count per block versus multiple rows. The overall count is the sum of the individual block counts. This approach is also highly parallelizable.

Why is Perl used in the Genezzo MicroKernel?

Perl was arrived at not by selection, but by a process of elimination of all other modern programing languages.

- **Garbage-Collection and No Pointers** - From the beginning, Genezzo

goals suggested strong memory management that enhances programmer-productivity, which eliminates C and C++
- **Dynamic Loading and Unloading with Zero Downtime to DB** - Difficult to implement in a managable way in C/C++, and shared library mechanisms are not built into the language.
- **Strong Language for System Programming** - Viable access to OS primitives but in a semi-portable way, such as a POSIX interface to operating system services. This requirement eliminated Java.
- **Wealth of Reusable Components** - A tremendous amount of Perl packages are available in CPAN, the *C*omprehensive *P*erl *A*rchive *N*etwork. A side benefit of this neutral/independent distribution mechanism is allowing the exchange of ideas and work by users of Genezzo. This advantage eliminates otherwise promising prospects of Python and Ruby.
- **Data Processing** - Suggests filtering and matching strength of Perl's well-known string processing abilities.

Where is Genezzo headed?

Near-term: innovation in databases

Long-term: Big picture view to extensibility and application efficiency, which will mandate application generation.

Programmer productivity will ultimately be the reason Genezzo can win over raw execution speed. If scalability and performance are issues, Genezzo will approach the problem by depending on the existence of larger and larger arrays of cheaper and cheaper hardware to run on and enjoy the benefit of parallelism. More processors will be the only way we see long term scalability in a world in which memory, network bandwidth, and storage are increasingly cheaper. Even if Moore's Law continues to hold for a long time, affordable processors for enterprise applications will continue to have moderate performance, and they will be added in small increments, allowing "scaling out" of clusters, rather than "scaling up" by replacing large individual servers.

How will Genezzo Systems, Inc. make money?

We feel great things will occur by the creation of the Genezzo technology. The exact business model of Genezzo Systems, however, is not public at this time. It should be noted, however, that Genezzo is not narrowly focused on selling software licenses for supported versions of open-source systems. While this may be one way in which Genezzo can provide utility in exchange for sustaining revenue, there are many new opportunities created by the use of Genezzo in the cost-effective construction of modern enterprise-class software. For more information about the services Genezzo provides and the application projects in the works, please contact info@genezzo.com .

Interesting Quotes

eWeek

Unfortunately, database technology has hit a plateau. The best vendors can do is increase speeds and attempt to extend the relational model to include user-definable extensions that will allow it to prevail for the foreseeable future. Technologies for parallelism and clustering are necessary and technically sophisticated, but they're lacking in vision.

The big question is if we're ready for the new wave of applications. First, in the medical field, miniature cameras are taking megabyte-size pictures every few seconds in millions of patients. Radio frequency identification tags are beginning to include time, geospatial data and more in real time. Billions of digital images are taken every year. But where are these images and data stored? They are in the file system, not the database.

The database is supposed to be able to handle, compare and become the platform for analytical applications for this data. But it's not going to happen any time soon—the relational model is stuck.

--John Taschek, eWeek Columnist
February 24, 2003
How the World Will Change Databases

Gartner

Gartner Study Confirms RDBMS Decline in 2002

A Gartner study shows the 2002 worldwide relational database management system (RDBMS) market declined 7 percent. IBM was overall market-share leader, and Oracle maintained its lead in the combined Unix, Windows and Linux market.

--Colleen Graham & Kevin H. Strange
Gartner
20 May 2003

MetaGroup

"Although previously perceived as a nonfactor through the end of this decade, we now believe open source databases will begin to show significant usage (i.e., 3 percent to 5 percent) within corporate datacenters by 2006."

--Metagroup, 3/2003